# Data Mining in Hybrid Languages via ILP

Francesca A. Lisi
Dipartimento di Informatica, University of Bari, Italy
lisi@di.uniba.it

**Abstract**

We present an application of the hybrid language $\mathcal{AL}$-log to frequent pattern discovery by using methods and techniques of Inductive Logic Programming.

## 1 Introduction

*Data mining* is an application area arisen in the 1990s at the intersection of several different research fields, notably statistics, machine learning and databases, as soon as developments in sensing, communications and storage technologies made it possible to collect and store large collections of scientific and commercial data [8]. The abilities to analyze such data sets had not developed as fast. Research in data mining can be loosely defined as the study of methods, techniques and algorithms for finding models or patterns that are interesting or valuable in large data sets. The space of patterns if often infinite, and the enumeration of patterns involves some form of search in one such space. Practical computational constraints place severe limits on the subspace that can be explored by a data mining algorithm. The goal of data mining is either *prediction* or *description*. Prediction involves using some variables or fields in the database to predict unknown or future values of other variables of interest. Description focuses on finding human-interpretable patterns describing data. Among descriptive tasks, data summarization aims at the extraction of compact patterns that describe subsets of data. There are two classes of methods which represent taking horizontal (cases) and vertical (fields) slices of the data. In the former, one would like to produce summaries of subsets, e.g. producing sufficient statistics or logical conditions that hold for subsets. In the latter case, one would like to describe relations between fields. This class of methods is distinguished from the above in that rather than predicting the value of a specified field (e.g., classification) or grouping cases together (e.g. clustering) the goal is to find relations between fields. One common output of this vertical data summarization is called *frequent (association) patterns*. These patterns state that certain combinations of values occur in a given database with a support greater than a user-defined threshold.

*Hybrid systems* are knowledge representation and reasoning (KR&R) systems consisting of two or more subsystems dealing with distinct portions of a single knowledge base by means of specific reasoning procedures [9]. E.g., $\mathcal{AL}$-log [7] combines DATALOG [4] and $\mathcal{ALC}$ [20]. In $\mathcal{AL}$-log the interplay between the *relational* and *structural* subsystems allows the representation of and the reasoning with objects, properties of

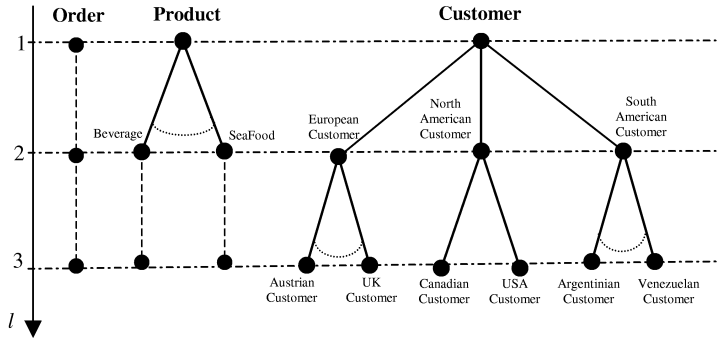


Figure 1: Concept hierarchies for the N$\frac{\text{ORTHWIN}}{\text{TRADERS}}$D database.

objects, relations between objects, and concept hierarchies. This is a great opportunity which has not been adequately exploited yet in applications. In this paper we propose $\mathcal{AL}$-log as a KR&R framework for data mining. Frequent pattern discovery at multiple levels of description granularity is the data mining task chosen as a showcase because it effectively shows the benefit of the $\mathcal{AL}$-log framework. From the methodological point of view, our work adopts techniques peculiar to Inductive Logic Programming (ILP), a research area at the intersection of machine learning and logic programming [18]. The usual KR&R framework in ILP is DATALOG. E.g., WARMR [6] is an ILP system for frequent pattern discovery where data and patterns are represented in DATALOG. We basically adjust the DL component of $\mathcal{AL}$-log in previous results obtained for DATALOG. Also we extend the unique names assumption to the relational component.

The paper is organized as follows. Section 2 defines the data mining task of interest. Section 3 presents the $\mathcal{AL}$-log framework. Section 4 discusses the ILP setting for working within the framework. Section 5 concludes the paper with final remarks and directions for future work.

## 2 The mining task

The data mining task chosen as a showcase is a variant of frequent pattern discovery which aims at taking concept hierarchies into account. This means that the problem statement includes some taxonomic information $\mathcal{T} = \{\mathcal{H}_k\}_{1 \leq k \leq m}$ besides the data set $\mathbf{r}$ to be mined. It is noteworthy that each concept hierarchy $\mathcal{H}_k$ in $\mathcal{T}$ can arrange its concepts $\{C_k^h\}_{1 \leq h \leq n_k}$ according to its own range of concept levels. Furthermore, data is typically available at leaf levels. This makes it hard to generate and evaluate patterns that combine concepts belonging to different hierarchies.

For the sake of uniformity, we map concepts to levels of description granularity whose number $maxG$ depends on the data mining problem at hand. Concepts marked with multiple granularity levels are simply replicated along the hierarchy they belong to, so that a layering of the taxonomy at hand is induced. We denote $\mathcal{T}^l$ the $l$-th layer, $1 \leq l \leq maxG$, of a taxonomy $\mathcal{T}$.

**Example 2.1.** In Figure 1 a three-layered taxonomy $\mathcal{T}$ is illustrated for the N$\frac{\text{ORTHWIN}}{\text{TRADERS}}$D

database distributed as a sample database for by Microsoft$^{TM}$ Access. We shall refer to it throughout this paper. It consists of two concept hierarchies, $\mathcal{H}_1$ and $\mathcal{H}_2$, rooted in `Product` and `Customer`, respectively. They have been rearranged according to the three problem-defined granularity levels. For instance, the concepts $\texttt{Beverage}, \ldots, \texttt{SeaFood}$ in $\mathcal{H}_1 = \{\texttt{Beverage} \sqsubset \texttt{Product}, \ldots, \texttt{SeaFood} \sqsubset \texttt{Product}\}$ have been assigned to both $\mathcal{T}^2$ and $\mathcal{T}^3$ to make the hierarchies balanced. $\Diamond$

A pattern is an expression in some language describing a subset of data or a model applicable to that subset [8]. Given a taxonomy $\mathcal{T}$, we denote by $\mathcal{L}^l$ the language of patterns involving concepts in $\mathcal{T}^l$.

**Definition 2.1.** Let $P \in \mathcal{L}^l$. A pattern $Q \in \mathcal{L}^h$, $h < l$ (resp. $h > l$), is an *ancestor* (resp. *descendant*) of $P$ iff it can be obtained from $P$ by replacing each concept $C$ that occurs in $P$ with a concept $D \in \mathcal{T}^h$ such that $C \sqsubseteq D$ (resp. $D \sqsubseteq C$). $\square$

This correspondence between $\mathcal{L}^l$ and $\mathcal{T}^l$ supplies means for getting both coarser-grained and finer-grained descriptions than a given pattern. Furthermore patterns are evaluated by taking their own level of description granularity into account. In particular, candidate patterns that fulfill the following requirements are retained:

**Definition 2.2.** Let $\mathbf{r}$ be a data set and $minsup^l$ the minimum support threshold for $\mathcal{L}^l$. A pattern $P \in \mathcal{L}^l$ with support $s$ is *frequent* in $\mathbf{r}$, denoted as $freq(\mathbf{r}, P)$, if (i) $s \geq minsup^l$ and (ii) all ancestors of $P$ w.r.t. $\mathcal{T}$ are frequent. $\square$

Formally, the problem of discovering frequent patterns at multiple levels of description granularity can be defined as follows.

**Definition 2.3.** Given

- a data set $\mathbf{r}$ including a taxonomy $\mathcal{T}$ where a reference concept and task-relevant concepts are designated,
- a set $\{\mathcal{L}^l\}_{1 \leq l \leq maxG}$ of languages
- a set $\{minsup^l\}_{1 \leq l \leq maxG}$ of support thresholds

the problem of *frequent pattern discovery at $l$ levels of description granularity*, $1 \leq l \leq maxG$, is to find the set $\mathcal{F}$ of all the patterns $P \in \mathcal{L}^l$ such that $freq(\mathbf{r}, P)$. $\square$

**Example 2.2.** An instantiation of Definition 2.3 for the case of N$\frac{\text{ORTHWIN}}{\text{TRADERS}}$D is sales analysis by finding associations between the category of ordered products and the geographic location of the customer within orders. Here `Order` is the reference concept, whereas the sub-concepts of `Product` and `Customer` are relevant to the task. $\Diamond$

Most algorithms of frequent pattern discovery follow the levelwise method for finding potentially interesting sentences of a given language [17]. Ingredients of this method are a data set $\mathbf{r}$, a language $\mathcal{L}$ of patterns, a generality relation $\succeq$ for $\mathcal{L}$, and a breadth-first search strategy for the space $(\mathcal{L}, \succeq)$ which alternates candidate generation and candidate evaluation phases. In our proposal the first two and the last two ingredients are supplied by the $\mathcal{AL}$-log framework (see Section 3) and the ILP setting (see Section 4), respectively.

# 3 The $\mathcal{AL}$-log framework

The main feature of our framework is the extension of the unique names assumption from the semantic level to the syntactic one. In particular we resort to the bias of Object Identity [16]: In a formula, terms denoted with different symbols must represent different entities of the domain. This bias leads to a restricted form of substitution whose bindings avoid the identification of terms: A substitution $\sigma$ is an *OI-substitution* w.r.t. a set of terms $T$ iff $\forall t_1, t_2 \in T: \ t_1 \neq t_2$ yields that $t_1\sigma \neq t_2\sigma$. We assume substitutions to be OI-compliant.

Data is represented as an *$\mathcal{AL}$-log knowledge base* $\mathcal{B}$, i.e. a pair $\langle \Sigma, \Pi \rangle$ where $\Sigma$ is an $\mathcal{ALC}$ knowledge base and $\Pi$ is a constrained DATALOG program. We remind that constraints in clauses are $\mathcal{ALC}$ concept assertions and the interaction between $\Sigma$ and $\Pi$ is at the basis of both a model-theoretic semantics and a hybrid reasoning mechanism (called *constrained SLD-resolution*) for $\mathcal{AL}$-log.

**Example 3.1.** Following Example 2.2, we consider an $\mathcal{AL}$-log knowledge base $\mathcal{B}$ obtained from the N$\frac{\text{ORTHWIN}}{\text{TRADERS}}$D database. To serve our illustrative purpose we focus on the concepts `Order`, `Product` and `Customer` and the relations *Order* and *OrderDetail*. The intensional part of $\Sigma$ encompasses inclusion statements such as `DairyProduct` $\sqsubset$ `Product` and `FrenchCustomer` $\sqsubset$ `EuropeanCustomer` that represent the two taxonomies illustrated in Figure 1. The extensional part of $\Sigma$ contains 830 concept assertions for `Order` (e.g. `order10248:Order`), 77 assertions for the sub-concepts of `Product`, e.g. `product11:DairyProduct`, and 91 assertions for the sub-concepts of `Customer`, e.g. `'VINET':FrenchCustomer`. The extensional part of $\Pi$ consists of 830 facts for `order`/14 and 2155 facts for `orderDetail`/5, e.g.

```
orderDetail(order10248,product11,'£14',12,0.00)
```

represents the order detail concerning the order number 10248 and product code 11. The intensional part of $\Pi$ defines two views on `order` and `orderDetail`:

```
item(OrderID,ProductID) ← orderDetail(OrderID,ProductID,_,_,_)
                          & OrderID:Order, ProductID:Product
purchaser(OrderID,CustomerID) ← order(OrderID,CustomerID,_,...,_)
                          & OrderID:Order, CustomerID:Customer
```

When triggered on the EDB of $\Pi$ these rules can deduce implicit facts such as `item(order10248,product11)` and `purchaser(order10248,'VINET')`. ◊

Patterns are represented as $\mathcal{O}$-queries, a rule-based form of unary conjunctive queries whose answer set contains individuals of an $\mathcal{ALC}$ concept $\hat{C}$ of reference.

**Definition 3.1.** Given a reference concept $\hat{C}$, an *$\mathcal{O}$-query* $Q$ to an $\mathcal{AL}$-log knowledge base $\mathcal{B}$ is a constrained DATALOG clause of the form

$$Q = q(X) \leftarrow \alpha_1, \ldots, \alpha_m \& X : \hat{C}, \gamma_2, \ldots, \gamma_n$$

where $X$ is the *distinguished variable* and the remaining variables occurring in the body of $Q$ are the *existential variables*. We denote by $key(Q)$ the key constraint $X : \hat{C}$ of $Q$. An $\mathcal{O}$-query $q(X) \leftarrow \& X : \hat{C}$ is called *trivial*. □

We impose $\mathcal{O}$-queries to be linked and connected (or range-restricted) constrained DATALOG clauses. The language $\mathcal{L}$ of patterns for a given mining problem is implicitly defined by a set $\mathcal{A}$ of atom templates, a key constraint $\hat{\gamma}$, and an additional set $\Gamma$ of constraint templates. An atom template $\alpha$ specify name and arity of the predicate and mode of its arguments. An instantiation of $\alpha$ is a DATALOG atom with predicate and arguments that fulfill the requirements specified in $\alpha$. Constraint templates specify the concept name for concept assertions and determine the level $l$ of granularity for descriptions.

**Example 3.2.** Following Example 3.1, let $\mathcal{A}$=`{item(+,-), purchaser(+,-)}` and $\hat{\gamma}$ be the key constraint built on the concept `Order`. Suppose that we are interested in descriptions at two different granularity levels. Thus the sets $\Gamma^1$ and $\Gamma^2$ of constraints are derived from the layers $\mathcal{T}^1$ and $\mathcal{T}^2$ of the taxonomy $\mathcal{T}$ shown in Figure 1. Examples of $\mathcal{O}$-queries belonging to this language are:

$Q_0$= `q(X)` $\leftarrow$ & `X:Order`
$Q_1$= `q(X)` $\leftarrow$ `item(X,Y)` & `X:Order`
$Q_2$= `q(X)` $\leftarrow$ `purchaser(X,Y)` & `X:Order`
$Q_3$= `q(X)` $\leftarrow$ `item(X,Y)` & `X:Order`, `Y:Product`
$Q_4$= `q(X)` $\leftarrow$ `purchaser(X,Y)` & `X:Order`, `Y:Customer`
$Q_5$= `q(X)` $\leftarrow$ `item(X,Y)`, `item(X,Z)` & `X:Order`, `Y:Product`
$Q_6$= `q(X)` $\leftarrow$ `item(X,Y)`, `purchaser(X,Z)` & `X:Order`, `Y:Product`
$Q_7$= `q(X)` $\leftarrow$ `item(X,Y)`, `item(X,Z)` & `X:Order`, `Y:Product`, `Z:Product`
$Q_8$= `q(X)` $\leftarrow$ `item(X,Y)`, `purchaser(X,Z)` & `X:Order`, `Y:Product`, `Z:Customer`
$Q_9$= `q(X)` $\leftarrow$ `item(X,Y)` & `X:Order`, `Y:DairyProduct`
$Q_{10}$= `q(X)` $\leftarrow$ `purchaser(X,Y)` & `X:Order`, `Y:EuropeanCustomer`
$Q_{11}$= `q(X)` $\leftarrow$ `item(X,Y)`, `item(X,Z)` & `X:Order`, `Y:DairyProduct`
$Q_{12}$= `q(X)` $\leftarrow$ `item(X,Y)`, `purchaser(X,Z)` & `X:Order`, `Y:DairyProduct`
$Q_{13}$= `q(X)` $\leftarrow$ `item(X,Y)`, `item(X,Z)`
& `X:Order`, `Y:DairyProduct`, `Z:GrainsCereals`
$Q_{14}$= `q(X)` $\leftarrow$ `item(X,Y)`, `purchaser(X,Z)`
& `X:Order`, `Y:DairyProduct`, `Z:EuropeanCustomer`

Note that all of them are linked and connected. Furthermore, $Q_0$ and $Q_1$ are valid for both $\mathcal{L}^1$ and $\mathcal{L}^2$, queries from $Q_2$ to $Q_8$ belong to $\mathcal{L}^1$, and queries from $Q_9$ to $Q_{14}$ belong to $\mathcal{L}^2$. In particular, $Q_8$ is an ancestor of $Q_{14}$. ◊

An *answer* to an $\mathcal{O}$-query $Q$ is a ground substitution $\theta$ for the distinguished variable of $Q$. The aforementioned conditions of well-formedness guarantee that the evaluation of $\mathcal{O}$-queries is sound. Query answering is necessary for computing the support of patterns during the candidate evaluation phases.

**Definition 3.2.** Let $\mathcal{B}$ be a $\mathcal{AL}$-log knowledge base, $P \in \mathcal{L}^l$. The *support* of $P$ with respect to $\mathcal{B}$ is defined:

$$\sigma(P, \mathcal{B}) = \frac{\mid answerset(P, \mathcal{B}) \mid}{\mid answerset(\widehat{P}, \mathcal{B}) \mid}$$

where $\widehat{P}$ is the trivial $\mathcal{O}$-query $q(X) \leftarrow \&X : \hat{C}$ for $\mathcal{L}^l$. □

**Example 3.3.** A correct answer to $Q_0$, $Q_3$ and $Q_9$ w.r.t. $\mathcal{B}$ is the substitution $\theta$ =`{X/order10248}`. In total we have that $answerset(Q_0, \mathcal{B})$ contains 830 answers

(as many as the number of individuals for the concept `Order`), $answerset(Q_3, \mathcal{B})$ 830 answers as well (since the conditions in the body of $Q_3$ are not strong enough to filter the individuals of `Order`) and $answerset(Q_9, \mathcal{B})$ 303 answers. Therefore, $\sigma(Q_3, \mathcal{B}) = 100\%$ and $\sigma(Q_9, \mathcal{B}) = 36.5\%$. $\Diamond$

# 4 The ILP setting

The definition of a generality order for $\mathcal{O}$-queries can not disregard the nature of $\mathcal{O}$-queries as a special case of constrained DATALOG clauses as well as the availability of an $\mathcal{AL}$-log knowledge base with respect to which these $\mathcal{O}$-queries are to be evaluated. Generalized subsumption [3] has been introduced in ILP as a generality order for Horn clauses with respect to background knowledge. We propose to adapt it to our $\mathcal{AL}$-log framework as follows.

**Definition 4.1.** Let $Q$ be an $\mathcal{O}$-query, $\alpha$ a ground atom, and $\mathcal{J}$ an interpretation. We say that $Q$ *covers* $\alpha$ under $\mathcal{J}$ if there is a ground substitution $\theta$ for $Q$ ($Q\theta$ is ground) such that $body(Q)\theta$ is true under $\mathcal{J}$ and $head(Q)\theta = \alpha$. $\square$

**Definition 4.2.** Let $P$ and $Q$ be two $\mathcal{O}$-queries to an $\mathcal{AL}$-log knowledge base $\mathcal{B}$. We say that $P$ *$\mathcal{B}$-subsumes* $Q$ if for every model $\mathcal{J}$ of $\mathcal{B}$ and every ground atom $\alpha$ such that $Q$ covers $\alpha$ under $\mathcal{J}$, we have that $P$ covers $\alpha$ under $\mathcal{J}$. $\square$

We have defined a quasi-order $\succeq_{\mathcal{B}}$ for $\mathcal{O}$-queries on the basis of $\mathcal{B}$-subsumption and provided a decidable procedure to check $\succeq_{\mathcal{B}}$ on the basis of constrained SLD resolution [15]. Furthermore $\succeq_{\mathcal{B}}$ is monotonic with respect to support.

**Proposition 4.1.** [15] Let $P$ and $Q$ be two $\mathcal{O}$-queries to an $\mathcal{AL}$-log knowledge base $\mathcal{B}$. If $P \succeq_{\mathcal{B}} Q$ then $\sigma(P, \mathcal{B}) \geq \sigma(Q, \mathcal{B})$.

Quasi-ordered sets can be searched by refinement operators [18]. From Proposition 4.1 it follows that downward refinement operators are of greater help in the context of frequent pattern discovery. Indeed, they drive the search towards patterns with decreasing support and enable the early detection of infrequent patterns. Furthermore we are interested in operators for searching multi-level pattern spaces. To this aim, given two $\mathcal{ALC}$ constraints $\gamma_1 = t_1 : C$ and $\gamma_2 = t_2 : D$, we say that $\gamma_1$ is *at least as strong as* (resp. *stronger than*) $\gamma_2$, denoted as $\gamma_1 \succeq \gamma_2$ (resp. $\gamma_1 \succ \gamma_2$), iff $t_1 = t_2$ and $C \sqsubseteq D$ (resp. $C \sqsubset D$).

**Definition 4.3.** Let $maxD$ be the search depth bound, and $\mathcal{L} = \{\mathcal{L}^l\}_{1 \leq l \leq maxG}$ be a language of $\mathcal{O}$-queries. A (downward) refinement operator $\rho_{\mathcal{O}}$ for $(\mathcal{L}, \succeq_{\mathcal{B}})$ is defined such that, for a given $\mathcal{O}$-query

$$P = q(X) \leftarrow \alpha_1, \ldots, \alpha_m \& X : \hat{C}, \gamma_2, \ldots, \gamma_n$$

in $\mathcal{L}^l$, $l < maxG$, $m + n < maxD$, the set $\rho_{\mathcal{O}}(P)$ contains all $Q \in \mathcal{L}$ that can be obtained by applying one of the following refinement rules:

$\langle Atom \rangle$ $Q = q(X) \leftarrow \alpha_1, \ldots, \alpha_m, \alpha_{m+1} \& X : \hat{C}, \gamma_2, \ldots, \gamma_n$ where $\alpha_{m+1}$ is an instantiation of an atom template in $\mathcal{A}$ such that $\alpha_{m+1} \notin body(P)$.

$\langle Constr \rangle$ $Q = q(X) \leftarrow \alpha_1, \ldots, \alpha_m \& X : \hat{C}, \gamma_2, \ldots, \gamma_n, \gamma_{n+1}$ where $\gamma_{n+1}$ is an instantiation of a constraint template in $\Gamma^l$ such that $\gamma_{n+1}$ constrains an unconstrained variable in $body(P)$.

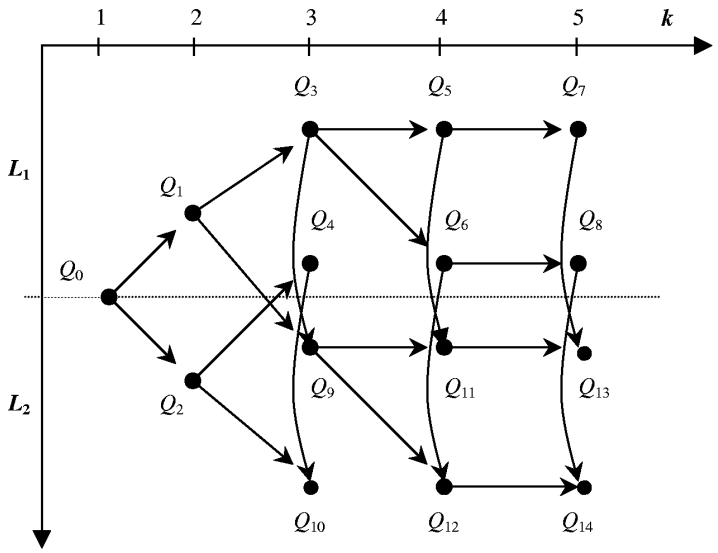


Figure 2: Portion of the refinement graph of $\rho_{\mathcal{O}}$ in $\mathcal{L}$.

$\langle \forall C \rangle$ $Q = q(X) \leftarrow \alpha_1, \ldots, \alpha_m \& X : \hat{C}, \gamma'_2, \ldots, \gamma'_n$ where each $\gamma'_j$, $2 \leq j \leq n$, is an instantiation of a constraint template in $\Gamma^{l+1}$ such that $\gamma'_j \succeq \gamma_j$ and at least one $\gamma'_j \succ \gamma_j$.

□

The rules $\langle Atom \rangle$ and $\langle Constr \rangle$ help moving within the pattern space $\mathcal{L}^l$ (*intra-space search*) whereas the rule $\langle \forall C \rangle$ helps moving from $\mathcal{L}^l$ to $\mathcal{L}^{l+1}$ (*inter-space search*). Both rules are correct, i.e. the $Q$'s obtained by applying any of these refinement rules to $P \in \mathcal{L}^l$ are $\mathcal{O}$-queries such that $P \succeq_{\mathcal{B}} Q$ [15].

**Example 4.1.** A refinement operator for $\mathcal{L}$ induces a refinement graph. This is a directed graph which has the members of $\mathcal{L}$ as nodes (here variant $\mathcal{O}$-queries can be viewed as the same node), and which contains an edge from $P$ to $Q$ just in case $Q \in \rho_{\mathcal{O}}(P)$. Figure 2 illustrates a portion of the refinement graph of $\rho_{\mathcal{O}}$ in the space $\mathcal{L} = \mathcal{L}^1 \cup \mathcal{L}^2$ of $\mathcal{O}$-queries reported in Example 3.2. Each edge indicates the application of only one of the rules defined for $\rho_{\mathcal{O}}$. E.g., $\rho_{\mathcal{O}}(Q_1)$ is the set

$Q'_1$= `q(X)` ← `item(X,Y)`, `item(X,Z)` & `X:Order`
$Q'_2$= `q(X)` ← `item(X,Y)`, `purchaser(X,Z)` & `X:Order`
$Q'_3$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Product`
$Q'_4$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Customer`
$Q'_5$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Beverage`
$Q'_6$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Condiment`
$Q'_7$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Confection`
$Q'_8$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:DairyProduct`
$Q'_9$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:GrainsCereals`
$Q'_{10}$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:MeatPoultry`
$Q'_{11}$= `q(X)` ← `item(X,Y)` & `X:Order`, `Y:Produce`

$Q'_{12}$= q(X) ← item(X,Y) & X:Order, Y:SeaFood
$Q'_{13}$= q(X) ← item(X,Y) & X:Order, Y:EuropeanCustomer
$Q'_{14}$= q(X) ← item(X,Y) & X:Order, Y:NorthAmericanCustomer
$Q'_{15}$= q(X) ← item(X,Y) & X:Order, Y:SouthAmericanCustomer

where $Q'_1$ and $Q'_2$ are generated by means of $\langle Atom \rangle$, $Q'_3$ and $Q'_4$ by means of $\langle Constr \rangle$, and the $\mathcal{O}$-queries from $Q'_5$ to $Q'_{15}$ also by means of $\langle Constr \rangle$ (but considering $Q_1$ as belonging to $\mathcal{L}^2$). Note that $Q'_4$, $Q'_{13}$, $Q'_{14}$, and $Q'_{15}$ will turn out to be infrequent. Yet they are generated. What matters while searching $(\mathcal{L}, \succeq_{\mathcal{B}})$ is to find patterns that are more specific than a given $P$ under $\mathcal{B}$-subsumption. Conversely, $\rho_{\mathcal{O}}(Q_3)$ is the set

$Q''_1$= q(X) ← item(X,Y), item(X,Z) & X:Order, Y:Product
$Q''_2$= q(X) ← item(X,Y), purchaser(X,Z) & X:Order, Y:Product
$Q''_3$= q(X) ← item(X,Y) & X:Order, Y:Beverage
$Q''_4$= q(X) ← item(X,Y) & X:Order, Y:Condiment
$Q''_5$= q(X) ← item(X,Y) & X:Order, Y:Confection
$Q''_6$= q(X) ← item(X,Y) & X:Order, Y:DairyProduct
$Q''_7$= q(X) ← item(X,Y) & X:Order, Y:GrainsCereals
$Q''_8$= q(X) ← item(X,Y) & X:Order, Y:MeatPoultry
$Q''_9$= q(X) ← item(X,Y) & X:Order, Y:Produce
$Q''_{10}$= q(X) ← item(X,Y) & X:Order, Y:SeaFood

where $Q''_1$ and $Q''_2$ are generated by means of $\langle Atom \rangle$, and the $\mathcal{O}$-queries from $Q''_3$ to $Q''_{10}$ by means of $\langle \forall C \rangle$. Note that $Q_9 = Q'_8 = Q''_6$ can be obtained by applying either $\langle Constr \rangle$ to $Q_1$ (here $Q_1$ is considered as belonging to $\mathcal{L}^2$) or $\langle \forall C \rangle$ to $Q_3$. ◊

From now on we call *k-patterns* those patterns $Q \in \rho^k_{\mathcal{O}}(P)$ that have been generated after $k$ refinement steps starting from the trivial $\mathcal{O}$-query $\widehat{P}$ in $\mathcal{L}^l$ and applying either $\langle Atom \rangle$ or $\langle Constr \rangle$. Under the assumption that $minsup^l \leq minsup^{l-1}$, $1 < l < maxG$, two pruning conditions for a multi-level search space can be defined.

**Proposition 4.2.** [15] Given an $\mathcal{AL}$-log knowledge base $\mathcal{B}$, a $k$-pattern $Q$ in $\mathcal{L}^l$ is infrequent if it is $\mathcal{B}$-subsumed by either (i) an infrequent $(k-1)$-pattern in $\mathcal{L}^l$ or (ii) an infrequent $k$-pattern in $\mathcal{L}^{l-1}$.

Condition (i) requires to test the containment in queries at the same description granularity level (*intra-space subsumption checks*) whereas condition (ii) demands for testing the containment in coarser-grained queries (*inter-space subsumption checks*). Because of Definition 2.2 the former are to be tested for each level $l$, while the latter only for $l > 1$.

**Example 4.2.** With reference to Figure 2, suppose now that $Q_9$ is a frequent pattern. It is refined into $Q_{11}$ by means of $\langle Atom \rangle$. If $Q_5$ was infrequent, $Q_{11}$ should be pruned according to Proposition 4.2(ii). ◊

Since each node in $\mathcal{L}^l$ can be reached from either another node in $\mathcal{L}^l$ or a node in $\mathcal{L}^{l-1}$ (as shown in Example 4.1), the pruning conditions of Proposition 4.2 allow us to speed up the search of spaces at levels $l > 1$ of description granularity [15].

## 5 Conclusions

Learning pure DLs has been quite widely investigated [5, 13, 2]. Conversely, there are very few attempts at learning in DL-based hybrid languages. In [19] the chosen language is CARIN-$\mathcal{ALN}$, therefore example coverage and subsumption between two hypotheses are based on the existential entailment algorithm of CARIN [14]. Following [19], Kietz studies the learnability of CARIN-$\mathcal{ALN}$, thus providing a pre-processing method which enables ILP systems to learn CARIN-$\mathcal{ALN}$ rules [12]. Closely related to DL-based hybrid systems are the proposals arising from the study of many-sorted logics, where a first-order language is combined with a sort language which can be regarded as an elementary DL [10]. In this respect the study of sorted downward refinement [11] can be also considered a contribution to learning in hybrid languages.

In this paper we have presented an application of hybrid languages to data mining. We would like to emphasize that the choice of an application context and the investigation of ILP issues within the chosen context make a substantial difference between our work and related work on learning in hybrid languages. Indeed our $\mathcal{AL}$-log framework can be seen as a simple yet significant object-relational data model. As such it has allowed us to bridge the gap between ILP and object-relational databases, thus paving the way to new interesting streams of research in ILP and data mining. Our ILP setting for *object-relational data mining* has been implemented in the ILP system $\mathcal{AL}$-QUIN ($\mathcal{AL}$-log QUery INduction) and tested on geographic and census data of Manchester Stockport (UK) [15].

In the future we plan to move from mining at multiple levels of granularity to mining at multiple levels of *abstraction* (aggregation/summarization) as usual in data warehousing. An extension of our work in this direction will require the investigation of the properties of DLs such as [1], and the definition of an ILP setting compliant with these properties.

## References


[1] F. Baader and U. Sattler. Description logics with concrete domains and aggregation. In H. Prade, editor, *Proc. of the 13th European Conference on Artificial Intelligence*, pages 336–340. John Wiley & Sons Ltd, 1998.

[2] L. Badea and S.-W. Nienhuys-Cheng. A refinement operator for description logics. In J. Cussens and A. Frisch, editors, *Inductive Logic Programming*, volume 1866 of *Lecture Notes in Artificial Intelligence*, pages 40–59. Springer-Verlag, 2000.

[3] W. Buntine. Generalized subsumption and its application to induction and redundancy. *Artificial Intelligence*, 36(2):149–176, 1988.

[4] S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer, 1990.

[5] W. Cohen and H. Hirsh. Learning the CLASSIC description logic: Thoretical and experimental results. In *Proc. of the 4th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'94)*, pages 121–133. Morgan Kaufmann, 1994.

[6] L. Dehaspe and H. Toivonen. Discovery of frequent DATALOG patterns. *Data Mining and Knowledge Discovery*, 3:7–36, 1999.

[7] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. $\mathcal{AL}$-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.

[8] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI Press/The MIT Press, 1996.

[9] A. Frisch and A. Cohn. Thoughts and afterthoughts on the 1988 workshop on principles of hybrid reasoning. *AI Magazine*, 11(5):84–87, 1991.

[10] A. Frisch. The substitutional framework for sorted deduction: Fundamental results on hybrid reasoning. *Artificial Intelligence*, 49:161–198, 1991.

[11] A. Frisch. Sorted downward refinement: Building background knowledge into a refinement operator for inductive logic programming. In S. Džeroski and P. Flach, editors, *Inductive Logic Programming*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 104–115. Springer, 1999.

[12] J.-U. Kietz. Learnability of description logic programs. In S. Matwin and C. Sammut, editors, *Inductive Logic Programming*, volume 2583 of *Lecture Notes in Artificial Intelligence*, pages 117–132. Springer, 2003.

[13] J.-U. Kietz and K. Morik. A polynomial approach to the constructive induction of structural knowledge. *Machine Learning*, 14(1):193–217, 1994.

[14] A. Levy and M.-C. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104:165–209, 1998.

[15] F.A. Lisi. *An ILP Setting for Object-Relational Data Mining*. Ph.D. Thesis, Department of Computer Science, University of Bari, Italy, 2002.

[16] F.A. Lisi, S. Ferilli, and N. Fanizzi. Object Identity as Search Bias for Pattern Spaces. In F. van Harmelen, editor, *Proc. of the 15th European Conference on Artificial Intelligence*, pages 375–379. IOS Press, 2002.

[17] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

[18] S.-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence*. Springer, 1997.

[19] C. Rouveirol and V. Ventos. Towards Learning in CARIN-$\mathcal{ALN}$. In J. Cussens and A. Frisch, editors, *Inductive Logic Programming*, volume 1866 of *Lecture Notes in Artificial Intelligence*, pages 191–208. Springer, 2000.

[20] M. Schmidt-Schauss and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.